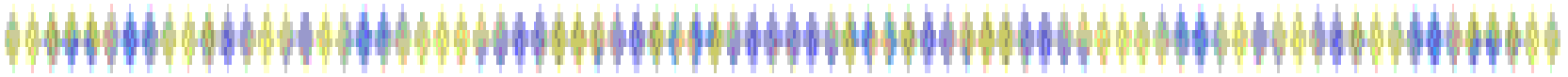
# 文書クラスタリングの技法ゼミ



10. TDTにおける単一パス・クラスタリング
11. 自己組織化マップの応用

発表日：2007/5/9

発表者：阿部 竜之介

# TDT

## TDTとは

- 何らかのイベントについての情報を、ニュースの文書などから成るコーパス（機械で処理できるような電子化テキスト資料）から自動識別する試み
- Topic Detection and Tracking

  Tracking ･･･ 教師付きの分類

  Detection ･･･ 教師なしのクラスタリング

- Detectionのために単一パス・アルゴリズムを使用

# Okapi方式（1）

◆クラスタと文書間の類似度計算

Okapi方式（確率型検索モデル）

文書$d_i$とクラスタ$C_k$の類似度は$C_k$に含まれる各文書との類似度の平均

$$s(d_i, C_k) = \frac{1}{\tilde{n}_k} \sum_{h:d_h \in C_k} s(\mathbf{d}_i, \mathbf{d}_h; C_k)$$

で計算される。

# Okapi方式（2）

●クラスタ$C_k$における文書$d_i$と文書$d_h$との類似度を

$$s(\mathbf{d}_i, \mathbf{d}_h; C_k) = \sum_{j=1}^{M} w_{ij} w_{hj} \times idf(t_j, C_k)$$

と定義

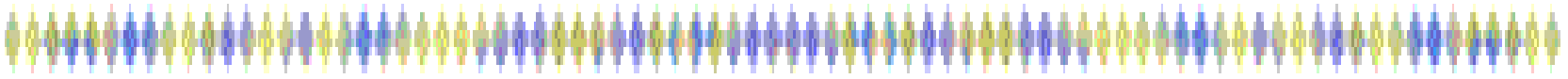
● $w_{ij}$は $\sum_{j=1}^{M} w_{ij} = 1$ となるように正規化したもの（$w_{hj}$も同様）。

# Okapi方式（3）

- $idf(t_j, C_k) = \log \frac{N - n_j + 0.5}{n_j + 0.5} + \lambda \frac{2n_{j|k}}{n_j + \tilde{n}_k}$

- $n_{j|k}$ はクラスタ $C_k$ に属する文書の中で、語 $t_j$ を含んでいる文書の数
- λ は定数

# TDTにおけるオンライン・クラスタリング



## オンライン・クラスタリングとは

- すでにクラスタリングされているところに、新たな文書が1件追加された場合に、それを既存のクラスタに割り当てるもの
- クラスタリングの方法は階層的クラスタ分析を使用
  - ただし、文書の到着順に1件ずつクラスタに割り当てていくという点で、かたちを変えた単一パス・アルゴリズムとして捉えることもできる
- 文書間類似度の計算にINQUERYにおける計算式

# INQUERYにおける計算式（1）

◆文書間の類似度計算

既存の文書を$\mathbf{d}_h$ 、新たに到着した文書を $\mathbf{d}_i$ とすると

$$s(\mathbf{d}_i, \mathbf{d}_h; \mathbf{d}_h) = \frac{\sum_{j=1}^{M} w_{hj} w_{ij}}{\sum_{J=1}^{M} w_{hj}}$$

で求められる

# INQUERYにおける計算式(2)

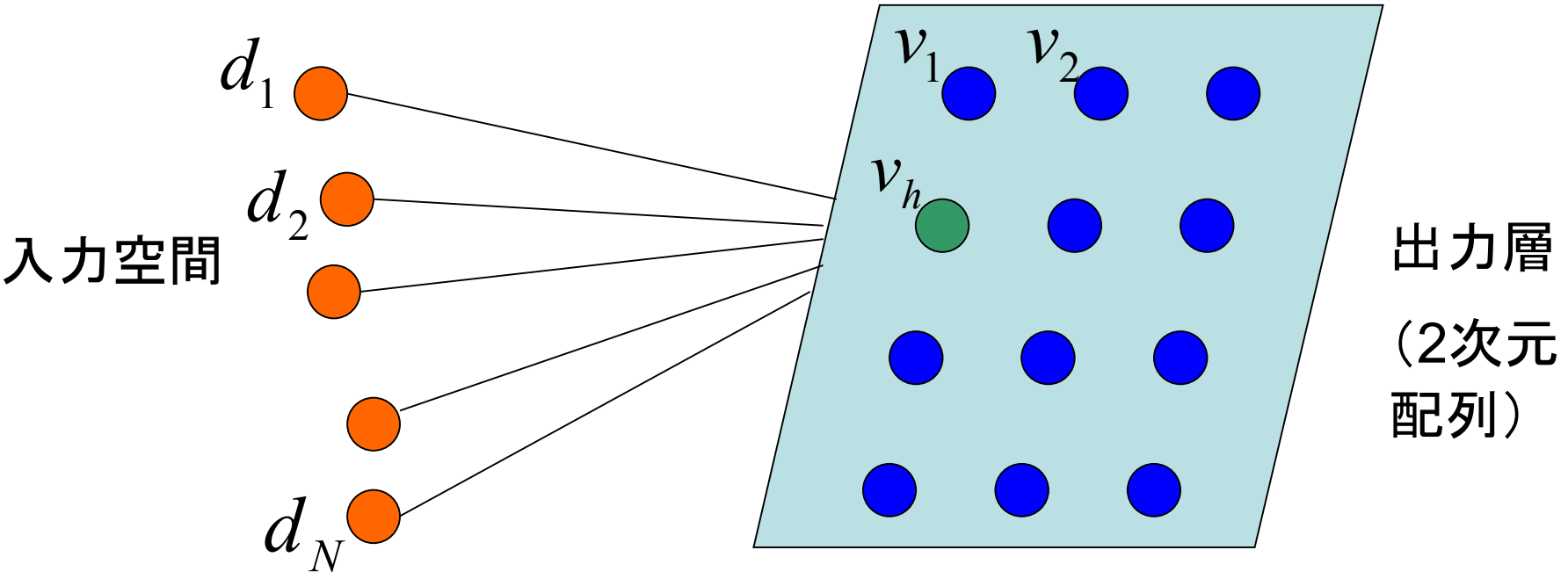
- $w_{ij} = 0.4 + 0.6 \times \dfrac{x_{ij}}{x_{ij} + 0.5 + 1.5 l_i / \tilde{l}} \times \dfrac{\log((N+0.5)/n_j)}{\log(N+1)}$

- $\tilde{l}$ は文書集合における文書長の平均
- $\tilde{l} = N^{-1} \sum_{i=1}^{N} l_i$ を意味する
- $w_{hj}$も同様

# 自己組織化マップ

## 自己組織化マップ（SOM）とは

- 1対の入力空間と出力層から構成される
- 一種のニューラルネットワーク



# SOMによる文書クラスタリングの例（1）

（1） 乱数を発生させ、出力ノードのベクトルを初期化する。

（2） 文書を順に入力する。文書ベクトルは入力ベクトルそのもの。

（3） 入力ベクトルと出力ベクトルとの距離を測り、勝者（最も近い）ノードを定義する（競合段階）。

（4） 勝者ノードとその周辺のノードがより入力ベクトルに近づくように調整する（協調段階）。

（5） （3）、（4）をG回繰り返す。

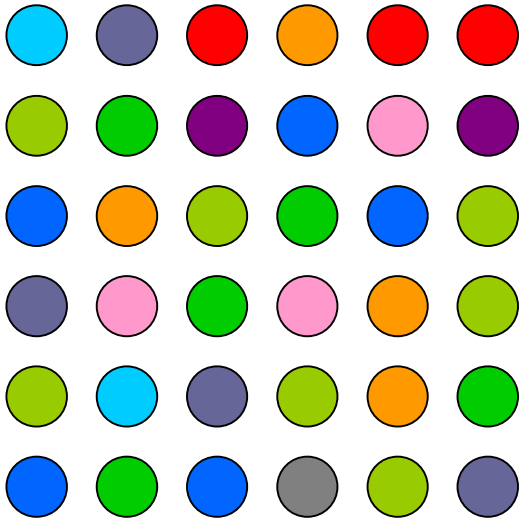
# SOMによる文書クラスタリングの例（2）

（6） 文書を順に入力し、最も距離の近い出力ノードにその文書を割り当てていく。

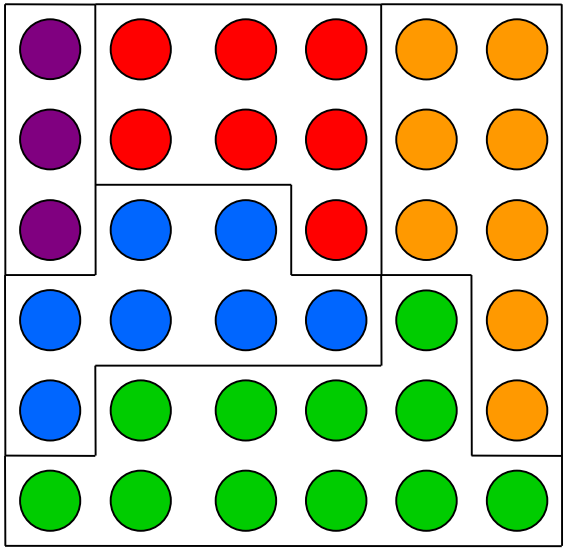
（7） 類似したノードをまとめて区域とし、それぞれの区域をクラスタとみなす。（具体的にどのように区域分けするかは不明）

（8）区域ごとに色分けすると非常に見やすい地図になる。

# SOMの色分け例

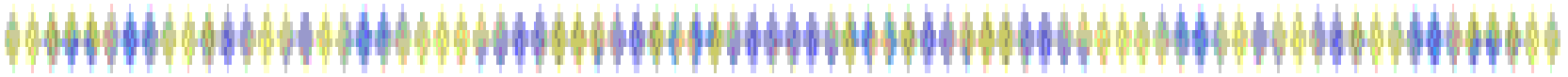


学習前



学習後

# SOM競合段階（1）



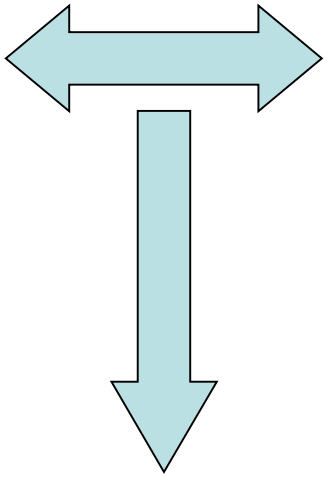
入力空間

出力層

文書ベクトル

$\mathbf{d}_i$

距離を測る



各出力ノードのベクトル

$\mathbf{v}_h \quad (h = 1,\ldots,H)$

$H$ 個の出力ノード

最も近い $h^*$ 番目を勝者ノードとして定義

$$\mathbf{v}_h = (v_{h1},\ldots,v_{hM})^T$$

# SOM競合段階（2）

◆近さをユークリッド距離の2乗で測る

$$h^* = \arg_h \min \left\| \mathbf{d}_i - \mathbf{v}_i \right\|^2$$

$$= \arg_h \min \sum_{j=1}^{M} (w_{ij} - v_{hj})^2$$

# SOM協調段階

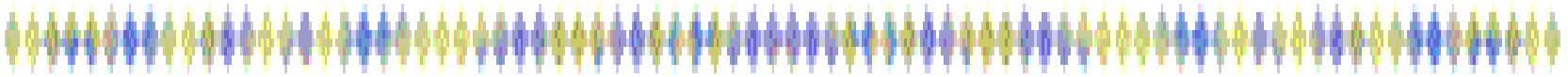
◆勝者ノードとその周辺のノードの重みを調整

- ●勝者ノードとその近傍の添字の集合を $N_{h^*}$
- $$v_{hj}^{(r+1)} = \begin{cases} v_{hj}^{(r)} + \eta^{(r)}(w_{ij} - v_{hj}^{(r)}), & h \in N_{h^*} \\ v_{hj}^{(r)}, & \text{それ以外} \end{cases}$$

$j = 1,\ldots,M$ について反復計算

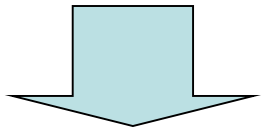
- $r$ は反復回数
- $\eta^{(r)}$ は学習率$(0 \le \eta^{(r)} \le 1)$、 $r$ 増加で減少

# WEBSOM（1）



·出力ノードの学習のためにかなりの数の反復が必要

×大規模なデータに対しては計算量が多くなる



**WEBSOM**

·大規模な文書集合に対してSOMを適用できるように工夫されている

# WEBSOM（2）

◆出力ノードのベクトル修正（協調段階）

$$v_{hj}^{(r+1)} = v_{hj}^{(r)} + f_{c(d_i),h}(r)(w_{ij} - v_{hj}^{(r)})$$

- $c(d_i)$は$d_i$ に対する勝者ベクトルの添字を返す関数

$$f_{c(d_i),h}(r) = \alpha(r)\exp(-\frac{\left\|t_h - t_{c(d_i)}\right\|^2}{2\sigma^2(r)})$$

- $t_h$ は出力領域中での位置を示す2次元ベクトル

# WEBSOM（3）

- $\alpha(r)$は $r$ 回目の反復における学習率
- $\sigma^2(r)$は、関数 $f_{c(d_i),h}(r)$ の広がりを調整するパラメータ

◆文書ベクトルの次元数を減らす変換

$$\tilde{\mathbf{d}}_i = \mathbf{B}\mathbf{d}_i$$

- 行列 $\mathbf{B}$ は $m \times M$ 行列（ただし、$m << M$ ）
- 行列 $\mathbf{B}$ は各列の要素が正規分布に従い、かつ長さが1になるように生成